128-bit addresses for the masses (of memory and devices).

Mathieu Bacou, Adam Chader, Chandana Deshpande, Christian Fabre, César Fuguet Tortolero, Pierre Michaud, Arthur Perais, Frédéric Pétrot, Gaël Thomas, Eduardo Tomasi Ribeiro

128-bit Addreses for the Masses (of Memory and Devices)

Maplurinum — ANR Project ANR-21-CE25-0016
Machinæ pluribus unum: One Machine out of Many

Christian Fabre — Univ. Grenoble Alpes, CEA, LIST, 38000 Grenoble, France

HOTINFRA' 23 — Workshop on Hot Topics in System Infrastructure
June 18, 2023, Orlando, Florida, USA — Co-located with ISCA 2023

Mathieu Bacou, Adam Chader (Télécom SudParis),
Chandana Deshpande (Grenoble INP), Christian Fabre,
César Fuguet (CEA List), Pierre Michaud (Inria),
Arthur Perais (CNRS), Frédéric Pétrot (Grenoble INP),
Gaël Thomas (Télécom SudParis), Eduardo Tomasi (CEA List)

Plan

1. **Why RISC-V 128 bit?**
2. Context & Vision
3. Some key 128 bit Issues
4. Operating System Research Avenues
5. System Architecture Research Avenues
6. µArchitecture Research Avenues
7. RISC-V 128 bit Software Toolset
8. Conclusion

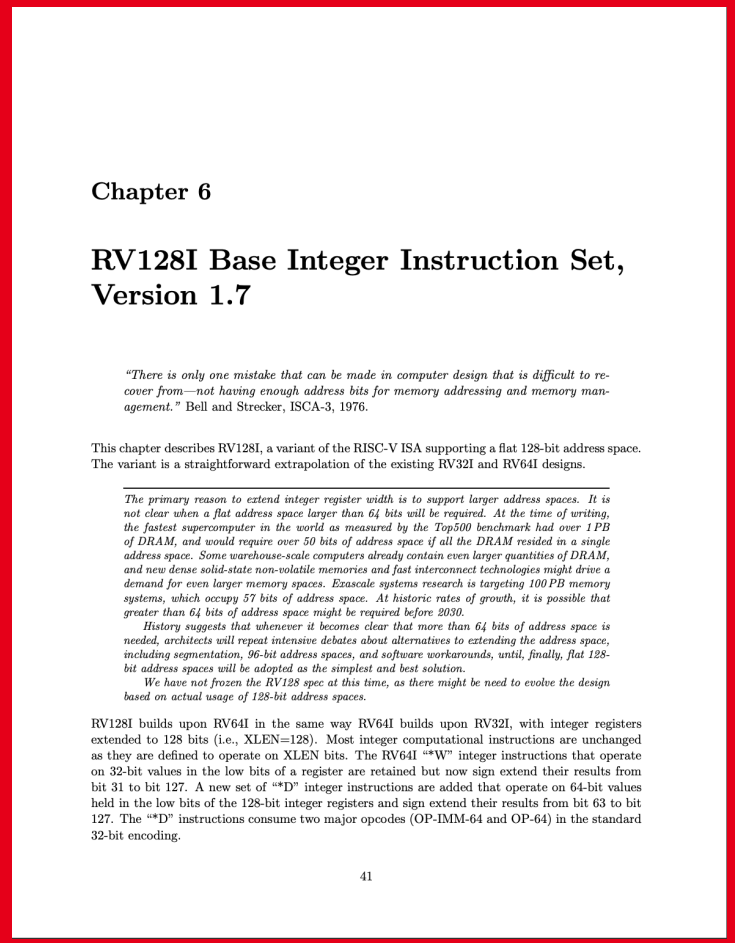
Why RISC-V 128 bit?

"There is only one mistake that can be made in computer design that is difficult to recover from — not having enough address bits for memory addressing and memory management."
Bell and Strecker, ISCA-3, 1976.

At historic rates of growth, it is possible that greater than 64 bits of address space might be required before 2030.

History suggests that whenever it becomes clear that more than 64 bits of address space is needed, architects will repeat intensive debates about alternatives to extending the address space, including segmentation, 96-bit address spaces, and software workarounds, until, finally, flat 128-bit address spaces will be adopted as the simplest and best solution.

We have not frozen the RV128 spec at this time, as there might be need to evolve the design based on actual usage of 128-bit address spaces.



Chapter 6

RV128I Base Integer Instruction Set, Version 1.7

"There is only one mistake that can be made in computer design that is difficult to recover from—not having enough address bits for memory addressing and memory management." Bell and Strecker, ISCA-3, 1976.

This chapter describes RV128I, a variant of the RISC-V ISA supporting a flat 128-bit address space. The variant is a straightforward extrapolation of the existing RV32I and RV64I designs.

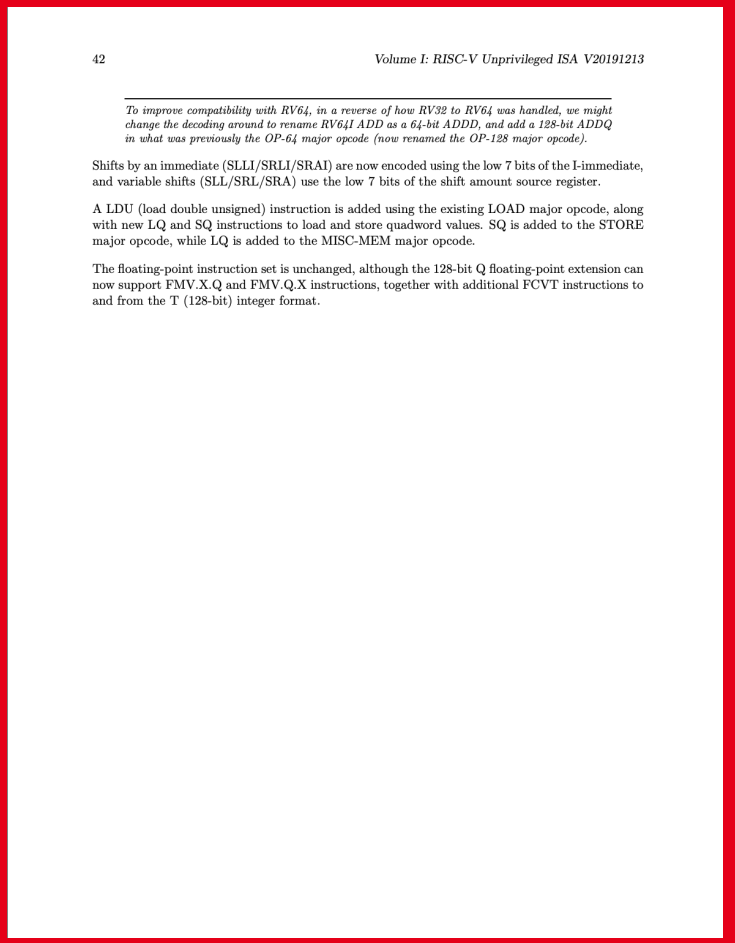
The primary reason to extend integer register width is to support larger address spaces. It is not clear when a flat address space larger than 64 bits will be required. At the time of writing, the fastest supercomputer in the world as measured by the Top500 benchmark had over 1 PB of DRAM, and would require over 50 bits of address space if all the DRAM resided in a single address space. Some warehouse-scale computers already contain even larger quantities of DRAM, and new dense solid-state non-volatile memories and fast interconnect technologies might drive a demand for even larger memory spaces. Exascale systems research is targeting 100 PB memory systems, which occupy 57 bits of address space. At historic rates of growth, it is possible that greater than 64 bits of address space might be required before 2030.

History suggests that whenever it becomes clear that more than 64 bits of address space is needed, architects will repeat intensive debates about alternatives to extending the address space, including segmentation, 96-bit address spaces, and software workarounds, until, finally, flat 128-bit address spaces will be adopted as the simplest and best solution.

We have not frozen the RV128 spec at this time, as there might be need to evolve the design based on actual usage of 128-bit address spaces.

RV128I builds upon RV64I in the same way RV64I builds upon RV32I, with integer registers extended to 128 bits (i.e., XLEN=128). Most integer computational instructions are unchanged as they are defined to operate on XLEN bits. The RV64I "*W" integer instructions that operate on 32-bit values in the low bits of a register are retained but now sign extend their results from bit 31 to bit 127. A new set of "*D" integer instructions are added that operate on 64-bit values held in the low bits of the 128-bit integer registers and sign extend their results from bit 63 to bit 127. The "*D" instructions consume two major opcodes (OP-IMM-64 and OP-64) in the standard 32-bit encoding.

41



42 *Volume I: RISC-V Unprivileged ISA V20191213*

To improve compatibility with RV64, in a reverse of how RV32 to RV64 was handled, we might change the decoding around to rename RV64I ADD as a 64-bit ADDD, and add a 128-bit ADDQ in what was previously the OP-64 major opcode (now renamed the OP-128 major opcode).

Shifts by an immediate (SLLI/SRLI/SRAI) are now encoded using the low 7 bits of the I-immediate, and variable shifts (SLL/SRL/SRA) use the low 7 bits of the shift amount source register.

A LDU (load double unsigned) instruction is added using the existing LOAD major opcode, along with new LQ and SQ instructions to load and store quadword values. SQ is added to the STORE major opcode, while LQ is added to the MISC-MEM major opcode.

The floating-point instruction set is unchanged, although the 128-bit Q floating-point extension can now support FMV.X.Q and FMV.Q.X instructions, together with additional FCVT instructions to and from the T (128-bit) integer format.

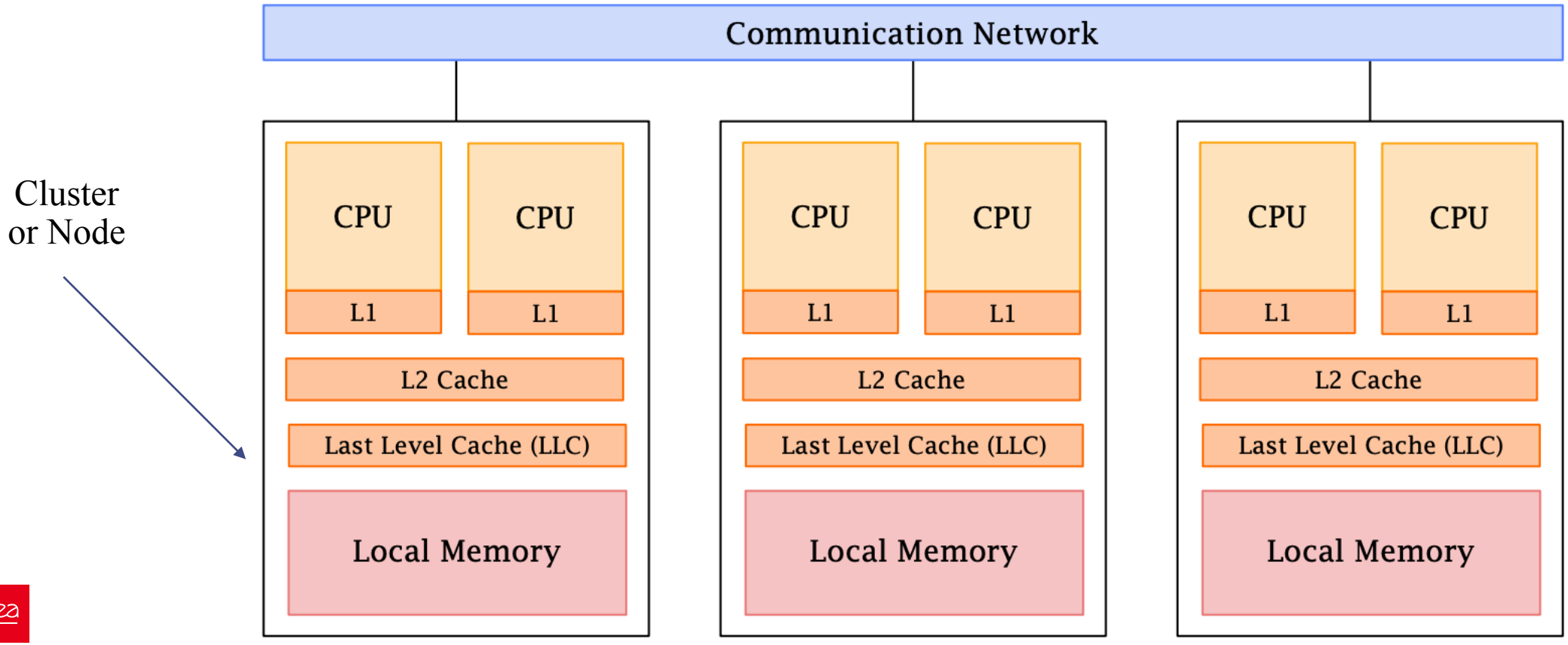
[1] A. Waterman and K. Asanović, "Chapter 6, RV128I Base Integer Instruction Set, Version 1.7," in The RISC-V Instruction Set Manual - Volume I: Unpriviliged ISA, 20191213, The RISC-V Foundation, 2019. Available online at `https://riscv.org/technical/specifications/`

Context & Vision (1/2)

Hight Performance Computing (HPC) Hardware View

floating-point dominated computations

Supercomputers are composed of multiple shared-memory nodes connected through a high-performance communication network.



Context & Vision (2/2)

- Regular increase of total system RAM on HPC
  - The 64→65 bit threshold might be passed sometime around 2035
- Advent of byte addressable NVRAM

- What is a 128 bit software playground?
  - Now: One Linux instance per shared memory domain + MPI
  - Then: Single System Image (SSI).
  - Virtual pointers are valid across the whole machine.

- Use cases
  - “HPC”: Distributing huge dataset over clusters, floating-point compute bound
  - “Datacenter” : Temporary mappings of many shared objects by many threads

- RISC-V and ISA extensions help mitigate heterogeneity
  - Clusters might not support the same set of extensions: vector vs ML vs variable precisions etc.

- Ideas take 10-20 years to percolate, so start early!

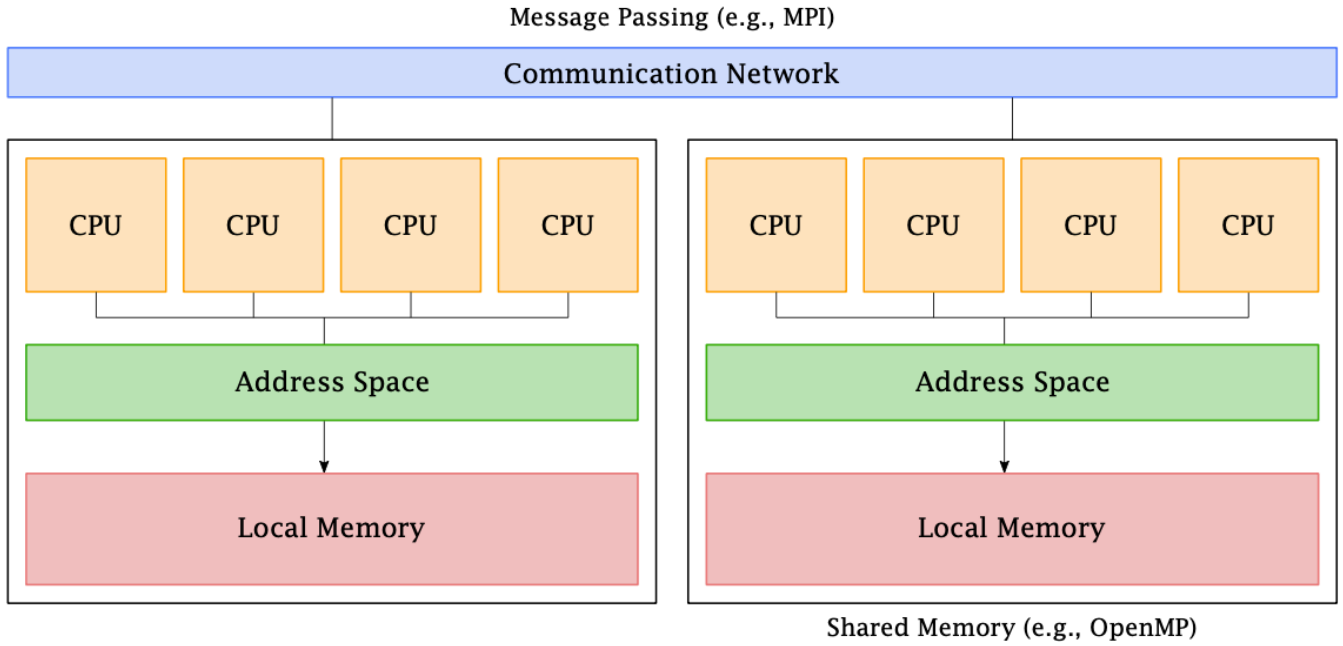
Some Key 128 bit Issues

- OS design:
  - Single kernel will not scale on millions of cores
  - How can user and processes see the whole machine as a single entity (SSI)?
    - What abstractions to expose?
  - Data migration from cluster to cluster by VM page remapping instead of message passing across virtual spaces
  - Someone has to manage data locality
- Software tooling
  - What would be the 128 bit C programming model?
- System architecture
  - Communication network
  - Memory coherency scope:
    - Full machine scope not achievable.
    - Shall we aim at cluster scope only?
  - Virtual memory at the machine level
- µarchitecture
  - HW complexity
  - Moore/Dennard laws are MIA, so moving to 128 bit will not be as easy as old' 32→64 bit transition

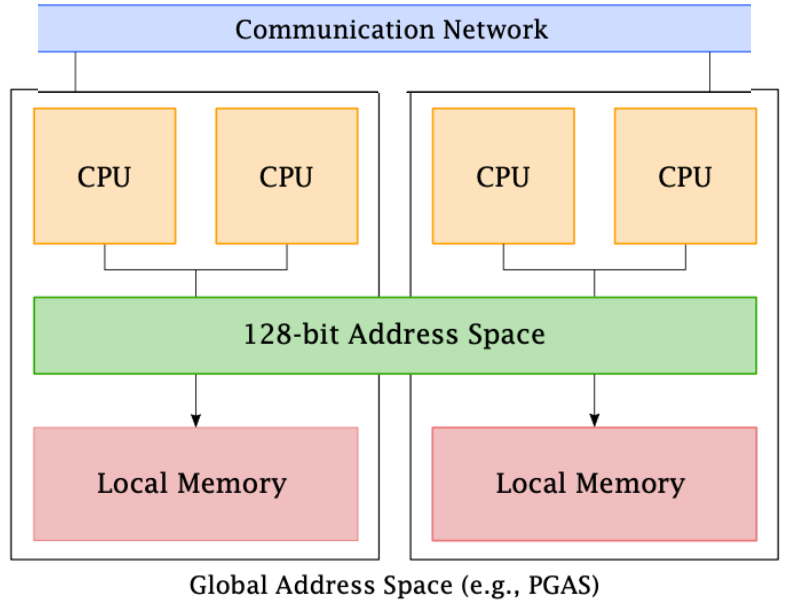
Plan

1. Why RISC-V 128 bit?
2. Context & Vision
3. Some key 128 bit Issues
4. **Operating System Research Avenues**
5. System Architecture Research Avenues
6. µArchitecture Research Avenues
7. RISC-V 128 bit Software Toolset
8. Conclusion

Operating System Research Avenues (1/2)



From one Linux per cluster
plus orchestration and middleware layers…



… to a machine-wide address space.
How to maintain and represent
such a virtual address space?

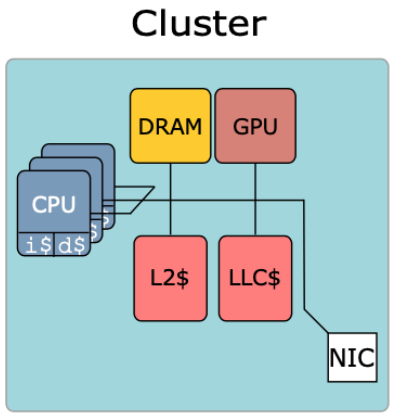
Operating System Research Avenues (2/2)

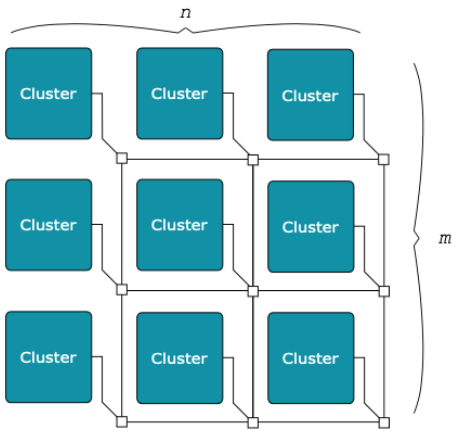
- Operating system at scale through multikernels:
  - One kernel per clusters
  - What is a virtual address space at the machine level?
  - Lend/borrow resources to other multikernels
- User abstractions
  - What is a machine-wide process?
  - Give view of topology to user, e.g. NUMA domains?
  - Allocate/free resources (does it need to be more than **`mmap()`** + *cpu bind* ?)
- Virtual Memory Open Questions
  - Page table organization:
    - More levels (business as usual) or novel organization?
    - Change minimum page size to 16K/ 32K or rely on superpages
  - How to ensure virtual address space consistency at the machine level
  - How to split HW/SW responsibilities for VM consistency
  - We do not have answers yet!

Plan

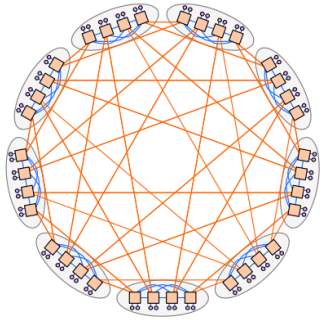
1. Why RISC-V 128 bit?
2. Context & Vision
3. Some key 128 bit Issues
4. Operating System Research Avenues
5. **System Architecture Research Avenues**
6. µArchitecture Research Avenues
7. RISC-V 128 bit Software Toolset
8. Conclusion

System Architecture Research Avenues

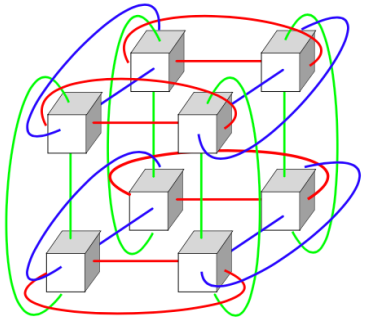




The cluster + interconnect architecture pattern is here to stay…



(a) Dragonfly



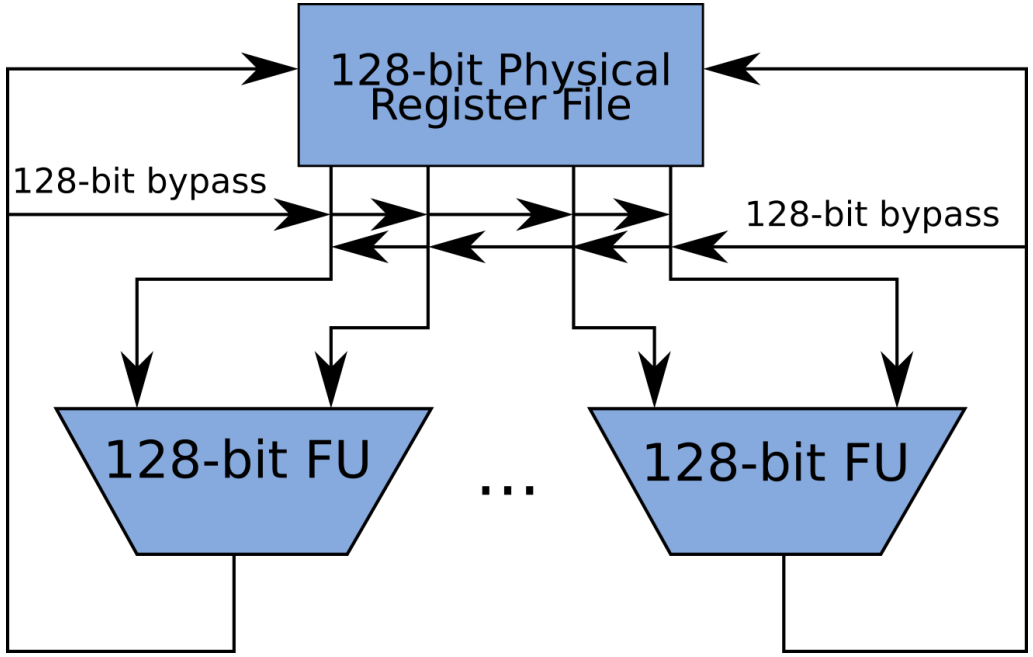
(b) Torus Fusion (Tofu)

…although the interconnect may change.

Plan

1. Why RISC-V 128 bit?
2. Context & Vision
3. Some key 128 bit Issues
4. Operating System Research Avenues
5. System Architecture Research Avenues
6. **µArchitecture Research Avenues**
7. RISC-V 128 bit Software Toolset
8. Conclusion

µArchitecure Research Avenues (1/3)



Some remarks:

- Implemented Virtual Addresses (VA) and Physical Addresses (PA) are not not 64 bit in current 64 bit HW
  - 128-bit transition is not 2x for µarchitecture structures that contain VA/PA, e.g. TLB tags, cache tags.
- Yet, registers, functional units, bypass network width has to double
  - This will bring latency, timing, and power costs
- Also, pointers occupy twice the space in cache memory
  - Should we increase cache line size to preserve spatial locality?

µArchitecure Research Avenues (2/3)

Problem statement — for now:

- Can we design a cost efficient transitional 128-bit µarchitecture with
  - Full 128-bit ISA support — i.e. RV128
  - Limited 128-bit data path
  - Reasonable performance

µArchitecure Research Avenues (3/4)

How we plan to address these issues?

- One key hypothesis: Get figures by compiling current C source for 128-bit.
  - `int` remains 32-bit, `long` remains 64-bit, etc.
  - Only pointer manipulation will be 128-bit
    - That is ~60% dynamic instructions on SPEC & Polybench on RV64G.
- Leverage this to divide & conquer the µarchitecture
  - 128-bit “address” cluster
  - 64-bit “compute” cluster
  - In essence, 64-bit instructions (40% dyn. insts.) will not consume 128-bit resources
- Address cluster has high value locality
  - Compress physical registers using region-based compression

Plan

1. Why RISC-V 128 bit?
2. Context & Vision
3. Some key 128 bit Issues
4. Operating System Research Avenues
5. System Architecture Research Avenues
6. µArchitecture Research Avenues
7. **RISC-V 128 bit Software Toolset**
8. Conclusion

RISC-V 128 bit Toolset

Data type	LLP128
char	8
short	16
int	32
long	64
long long	128
void *, size_t, ptrdiff_t	128

Frédéric Pétrot's Github page:
`https://github.com/fpetrot/riscv-gnu-toolchain`

Software tools operational on bare metal:

- QEMU **`-cpu=x-rv128`**
  - most of it upstreamed since jan 2022.
- Binutils + gas + gld + gdb:
  - ELF128 support
  - RV128 opcodes
  - 128-bit relocs
  - *Relatively* simple thanks to **`__[u]int128_t`** native support
- gcc:
  - Instructions in machine description file
  - Only **`long long`** and pointers are 128 bit wide
- Still full of bugs, but a first step!

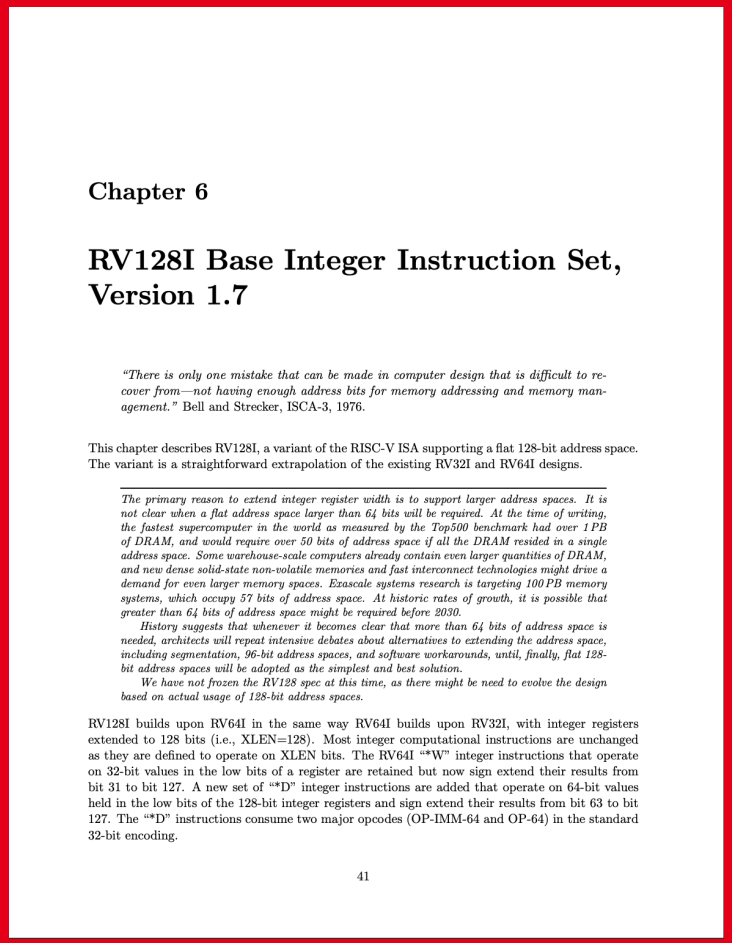
Conclusion: Let's define what RISC-V 128 bit will be!

"There is only one mistake that can be made in computer design that is difficult to recover from — not having enough address bits for memory addressing and memory management."
Bell and Strecker, ISCA-3, 1976.

At historic rates of growth, it is possible that greater than 64 bits of address space might be required before 2030.

History suggests that whenever it becomes clear that more than 64 bits of address space is needed, architects will repeat intensive debates about alternatives to extending the address space, including segmentation, 96-bit address spaces, and software workarounds, until, finally, flat 128-bit address spaces will be adopted as the simplest and best solution.

We have not frozen the RV128 spec at this time, as there might be need to evolve the design based on actual usage of 128-bit address spaces.



Chapter 6

RV128I Base Integer Instruction Set, Version 1.7

"There is only one mistake that can be made in computer design that is difficult to recover from—not having enough address bits for memory addressing and memory management." Bell and Strecker, ISCA-3, 1976.

This chapter describes RV128I, a variant of the RISC-V ISA supporting a flat 128-bit address space. The variant is a straightforward extrapolation of the existing RV32I and RV64I designs.

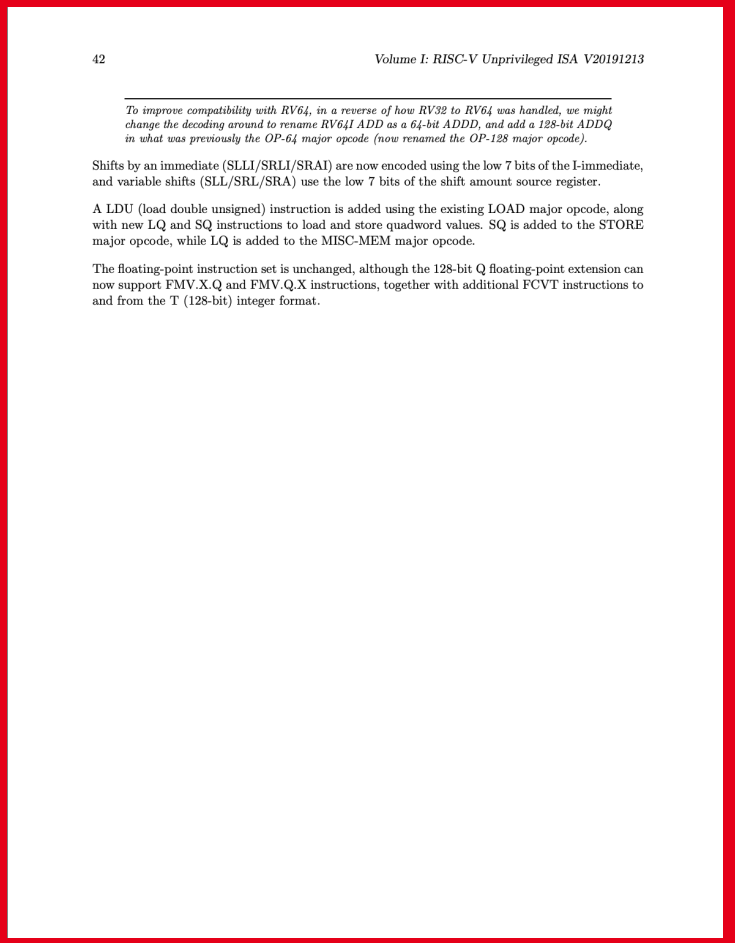
The primary reason to extend integer register width is to support larger address spaces. It is not clear when a flat address space larger than 64 bits will be required. At the time of writing, the fastest supercomputer in the world as measured by the Top500 benchmark had over 1 PB of DRAM, and would require over 50 bits of address space if all the DRAM resided in a single address space. Some warehouse-scale computers already contain even larger quantities of DRAM, and new dense solid-state non-volatile memories and fast interconnect technologies might drive a demand for even larger memory spaces. Exascale systems research is targeting 100 PB memory systems, which occupy 57 bits of address space. At historic rates of growth, it is possible that greater than 64 bits of address space might be required before 2030.

History suggests that whenever it becomes clear that more than 64 bits of address space is needed, architects will repeat intensive debates about alternatives to extending the address space, including segmentation, 96-bit address spaces, and software workarounds, until, finally, flat 128-bit address spaces will be adopted as the simplest and best solution.

We have not frozen the RV128 spec at this time, as there might be need to evolve the design based on actual usage of 128-bit address spaces.

RV128I builds upon RV64I in the same way RV64I builds upon RV32I, with integer registers extended to 128 bits (i.e., XLEN=128). Most integer computational instructions are unchanged as they are defined to operate on XLEN bits. The RV64I "*W" integer instructions that operate on 32-bit values in the low bits of a register are retained but now sign extend their results from bit 31 to bit 127. A new set of "*D" integer instructions are added that operate on 64-bit values held in the low bits of the 128-bit integer registers and sign extend their results from bit 63 to bit 127. The "*D" instructions consume two major opcodes (OP-IMM-64 and OP-64) in the standard 32-bit encoding.

41



42 Volume I: RISC-V Unprivileged ISA V20191213

To improve compatibility with RV64, in a reverse of how RV32 to RV64 was handled, we might change the decoding around to rename RV64I ADD as a 64-bit ADDD, and add a 128-bit ADDQ in what was previously the OP-64 major opcode (now renamed the OP-128 major opcode).

Shifts by an immediate (SLLI/SRLI/SRAI) are now encoded using the low 7 bits of the I-immediate, and variable shifts (SLL/SRL/SRA) use the low 7 bits of the shift amount source register.

A LDU (load double unsigned) instruction is added using the existing LOAD major opcode, along with new LQ and SQ instructions to load and store quadword values. SQ is added to the STORE major opcode, while LQ is added to the MISC-MEM major opcode.

The floating-point instruction set is unchanged, although the 128-bit Q floating-point extension can now support FMV.X.Q and FMV.Q.X instructions, together with additional FCVT instructions to and from the T (128-bit) integer format.

[1] A. Waterman and K. Asanović, "Chapter 6, RV128I Base Integer Instruction Set, Version 1.7," in The RISC-V Instruction Set Manual - Volume I: Unpriviliged ISA, 20191213, The RISC-V Foundation, 2019. Available online at `https://riscv.org/technical/specifications/`


Thanks a lot!
Questions?
Mathieu Bacou, Adam Chader (Télécom SudParis),
Chandana Deshpande (Grenoble INP), Christian Fabre,
César Fuguet (CEA List), Pierre Michaud (Inria),
Arthur Perais (CNRS), Frédéric Pétrot (Grenoble INP),
Gaël Thomas (Télécom SudParis), Eduardo Tomasi (CEA List)
Maplurinum — ANR Project ANR-21-CE25-0016
Machinæ pluribus unum: One Machine out of Many
© Samuel Blanc
www.horizonspartages.fr